ARL-TR-7602 ● FEB 2016

US Army Research Laboratory

Computer Modeling of the Effects of Atmospheric Conditions on Sound Signatures

by Sarah Wagner, Adrienne Raglin, and John Noble

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-7602 ● FEB 2016

US Army Research Laboratory

Computer Modeling of the Effects of Atmospheric Conditions on Sound Signatures

by Sarah Wagner
Science and Engineering Apprentice Program (SEAP), George Washington University

Adrienne Raglin and John Noble
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
February 2016	Final	May 2014–August 2015

4. TITLE AND SUBTITLE
Computer Modeling of the Effects of Atmospheric Conditions on Sound Signatures

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Sarah Wagner, Adrienne Raglin, and John Noble

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CIE-S
2800 Powder Mill Road
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-7602

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The goals of the project were to analyze the effects of atmospheric conditions on sound propagation, create a filter to model effects under different conditions, and apply the filter to sound produced by unmanned aircraft systems or to any other propagating sound. The new sound files produced could be used in various simulations and provide a better understanding of atmospheric effects. First, the Scanning Fast Field Program was used to obtain decibel attenuations at different frequencies under various conditions. Second, those data were imported into MATLAB and used to construct a filter to attenuate a chosen sound file. Third, the graphic user interface (GUI) developed under this project allows the selection of various atmospheric and geometric conditions, the upload of a sound file, and then generates output of the modified sound. The GUI presents the graph of the filter and the changes in magnitude for the original and modified data. An added feature of the GUI allows the changes to the modified sound to be removed, restoring the original data. In the future, the GUI could be expanded to account for physical atmospheric effects on the microphone potentially improving the accuracy of the model.

15. SUBJECT TERMS
sound signatures, atmospheric conditions, sound modification, sound propagation, filters

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Adrienne Raglin
Unclassified	Unclassified	Unclassified	UU	32	19b. TELEPHONE NUMBER (Include area code) 301-394-0210

Contents

List of Figures

List of Tables

1. Introduction/Background

Unmanned aircraft systems (UASs) are being used by the general public, industry, academia, and the government. One critical challenge is detecting UASs that may pose a possible threat. Small or obscured UASs may be the most difficult to detect. The audible signal from some UASs may be a cue for detection. For this application, it would be useful to recognize UAS audio sound signatures under different conditions. The ability to accurately retrieve the original signal from data with distortions due to the atmosphere may also aid in the detection process. For this project, a program to simulate distorted sound under any set of initial conditions was designed and implemented.

Conditions that may affect sound propagation include the temperature gradient, the humidity of the air, the cloud cover, the wind speed and the direction, and the location of both the source and the receiver.[1] Wind can carry sound longer distances if it moves in the same direction as the sound signal. The humidity of the air contributes to the elasticity of the air, which affects the speed of sound. Sound also travels faster through cool air, so at night sound travels farther closer to the ground as opposed to refracting up into the atmosphere.[2] These modifying conditions affect the various frequencies in a sound signature differently and therefore each frequency in a sound signature will be modified differently for each set of environmental conditions.

The Scanning Fast Field Program (SCAFFIP) was designed to receive various parameters as initial conditions and to predict the decibel attenuations on certain discrete frequencies. SCAFFIP matches the initial conditions to certain atmospheric profiles already in the database and uses that combination of data to generate a layer-by-layer description of the atmosphere. The atmospheric propagation effect at different distances is then generated as a matrix of decibel attenuations.[3]

The experiment conducted for this project uses those discrete decibel predictions to produce the sound that would be heard at a distance under the input conditions and outputs that new sound using a graphic user interface (GUI). Data were also collected to test the simulation. The results from this work could be used to improve understanding of the effect of attenuation of certain frequencies on a sound signature that can be used to recognize distorted sounds of different UASs. Future applications could be developed to more accurately detect sounds that originate at long distances.

2. Experiment/Calculations

2.1 Simulate Atmospheric Distortion

To model the effects of the atmosphere on sound waveforms, a filter was constructed to mimic atmospheric sound attenuation. This filter was constructed based on the SCAFFIP. SCAFFIP considered information regarding the longitude and latitude of the location, the time of day, the estimated cloud cover, the day of the year, the wind speed and wind direction, the temperature, the relative humidity, the height of the source, and the height of the receiver. SCAFFIP also took in parameters defining the distances and the frequencies for calculating the attenuation. With that information, SCAFFIP then generated a matrix of decibel attenuations for specific frequencies at specific distances.

The sound file was read into MATLAB as a vector quantity of amplitude versus time. That vector was analyzed using a Discrete Fourier Transform, which generated a new vector of amplitude versus frequency, allowing the sound to be analyzed in terms of frequency (see Appendix: dftFreq). The SCAFFIP-generated matrix was linearly interpolated to provide a decibel attenuation for every frequency present in the transformed sound file. Because SCAFFIP only predicted for lower frequency ranges, higher frequency ranges were approximated based on humidity and distance (see Appendix: Function dBVec).

To maintain reversibility of the function, the phase and the magnitude of the transformed sound file were saved separately. The filter was applied to the magnitude of the sound, using the following relationship:

$$dB=20*\log(value/ref), \tag{1}$$

where *dB* is the decibel attenuation; *value* is the new sound amplitude; and *ref* is the reference, or original sound amplitude.

The relationship solved for the value gives the following:

$$value= ref*10\text{^}(dB/20). \tag{2}$$

The new magnitude of the transformed sound with the phase was obtained using Euler's Formula:

$$|z|\ (\cos\phi + i\sin\phi) = amp, \tag{3}$$

where $|z|$ was the magnitude after the filter was applied, ϕ was the phase, and *amp* was the new amplitude versus frequency vector. An inverse Fourier transform was

applied to the new amplitude to revert the sound signal back to amplitude versus time vector. This new amplitude versus time vector was the distorted sound (see Appendix: getDistortFFT and audioDistort). To reverse the distortion and recover an original sound file from collected data, the filter was inverted. To identify the distance between an original and a distorted sound file, the decibel attenuation at each discrete frequency that SCAFFIP calculated was evaluated using Eq. 1, and that profile was matched to a distance in SCAFFIP's frequency versus distance matrix, minimizing the square of the error (see Appendix: findDistance).

These effects were displayed in a GUI. The GUI was developed in MATLAB using the GUI guide tool. It used the audio of the distorted sound in addition to graphs of the waveform of the sound and plotted its frequency versus magnitude to display atmospheric effects on the sound file.

2.2 Data Collection

Sample distorted sounds were collected under known conditions to test the simulation. The environmental data collected included the longitude and the latitude of the location, the time of day, the estimated cloud cover, the day of the year, the wind speed and wind direction, the temperature, and the relative humidity. The geometric data collected included the height of the source and the height of the receiver.

The longitude and the latitude were determined using a global positioning system (GPS) with an accuracy of one decimal place. The time of day was recorded with a digital 24-h clock to an accuracy of 1 h. The percentage of cloud cover was estimated by eye to the nearest multiple of 10. The wind speed was collected in miles per hour using an anemometer to an accuracy of 2 decimal places. The anemometer was used to determine the wind direction, and a compass was used to refine the measurement of the direction to an accuracy of a second. The temperature was collected in Celsius using a thermometer accurate to 2 decimal places. The relative humidity was collected using a hygrometer accurate to 2 decimal places.

2.3 Gathering Sound

The sound source, a 30-s sound clip, was set up at the start point. The sound source was recorded at an audio sample rate of 44 kHz with 2 channels at a bit rate of 114 kbps. A straight line out from the sound source was marked using a piece of string. A distance of 5 m was measured out using a measuring tape. The sound was recorded at the source and at 5-m intervals to 60 m.

3. Results

The GUI (Fig. 1) was developed to allow the user to upload a sound file in the *.wav format, to input atmospheric conditions, and to view characteristics of the original and distorted sound. For the program to function, the user must input the atmospheric conditions into SCAFFIP to generate the proper filter files, which will then be imported to the program.

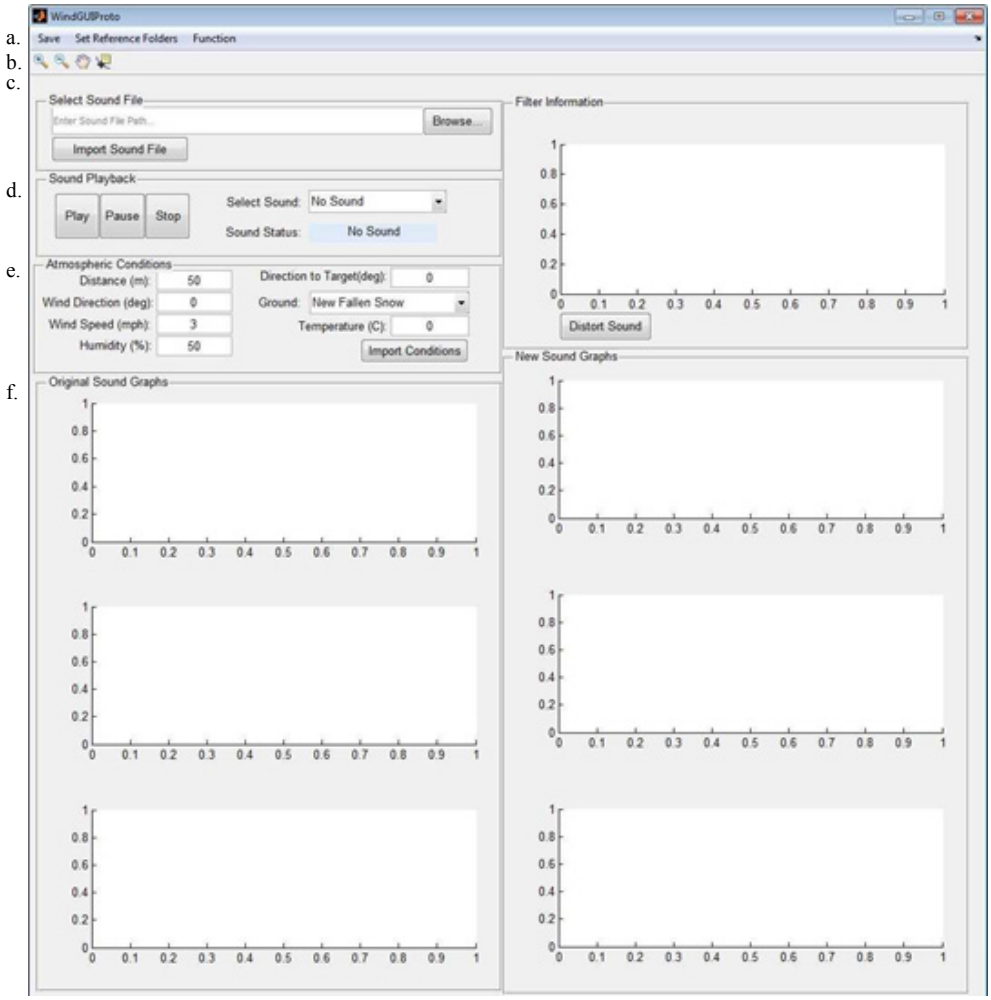

Fig. 1 Blank template of graphic user interface

3.1 Original Sound Display Functionalities

The user selects a file to upload in the Select Sound File field (Fig. 1c) and can either type the path to the desired *.wav file or the user can use the browse button to find the path. The file to upload must be a *.wav file.

The user then imports the sound file using the button labeled Import Sound File. This turns the path text black and makes sound playback for the original sound available.

Three graphs are generated on the GUI as shown in Fig. 1f. The first graph gives the amplitude of the sound signal over time. The second graph shows the magnitude in the signal of each different frequency in Hertz, but only shows the lower frequencies. The third graph gives a logarithmically scaled view of the magnitudes of the different frequencies (Fig. 2.)

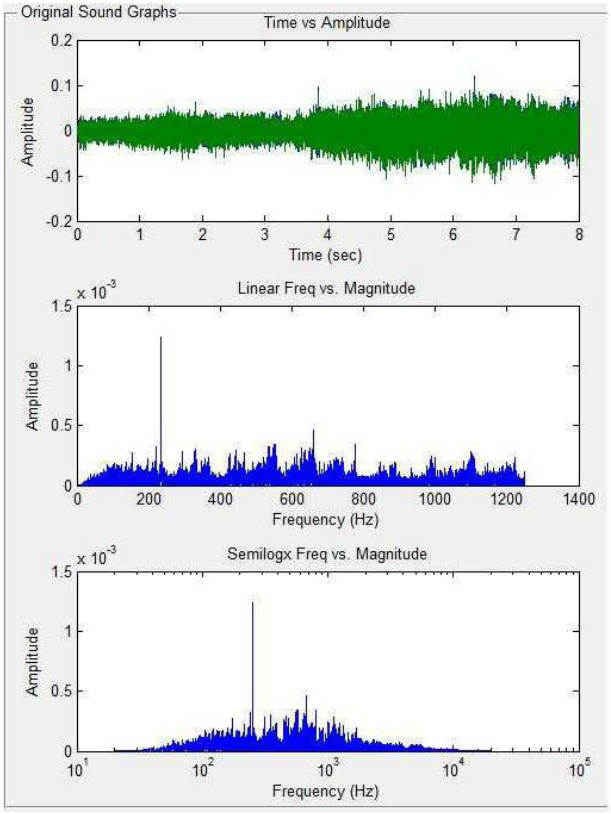

Fig. 2 Original sound graphs. First graph (top) displays time vs. amplitude of the signal coordinates. The second (middle) displays the frequency vs. the magnitude of the signal. The third (bottom) displays the logarithm of the frequency vs. the magnitude of the signal.

The user can pan, zoom, and acquire exact data points on any of the graphs using symbols on the toolbar (Fig. 1b).

3.2 Filter Display Functionalities

The user inputs the filter information in the field labeled Atmospheric Conditions (Fig. 1e). The user should also run SCAFFIP under the desired conditions to create the necessary *.ptd files.

The user can choose between 9 different ground conditions: New Fallen Snow, 2-Layer Snow, Sugar Snow, Forest Floor, Grass Covered Pasture, Roadside Dirt, Packed Sandy Silt, Exposed/Rain-Packed Dirt, or Asphalt Covered. The user can also enter the relative humidity of the environment, information about the wind profile, the temperature, and the distance from the sound receiver to the sound source. The distance cannot exceed the range calculated by SCAFFIP, which is 80 m. The user imports the filter generated by SCAFFIP using the import button. This will generate a graph of the decibel attenuations (example shown in Fig. 3) in the field labeled Filter Information (Fig. 1g).

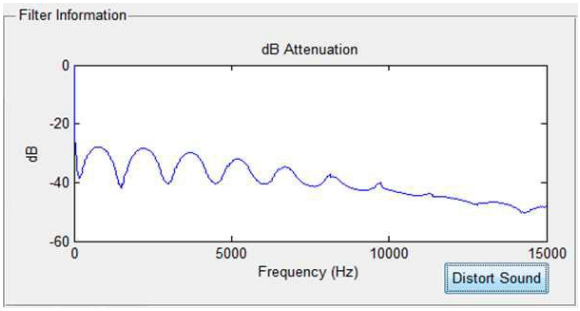

Fig. 3 Decibel attenuations of each frequency that are predicted for the given atmospheric conditions

3.3 Distort Sound

The Distort Sound button generates the 3 graphs on the right side of the GUI (Fig. 1h). These 3 are the same types of graphs as the graphs generated for the original sound file and are presented for comparison. Sound playback for the distorted sound is now available in the field labeled Sound Playback (Fig. 1d).

The user can play, pause, or stop whatever sound is selected using the Play, Pause, and Stop buttons (Fig. 1d).

The blue box labeled Sound Status will indicate the current state of the sound selected (Fig. 1d). No Sound indicates no sound has been uploaded yet. Done indicates the sound has finished uploading or playing and is ready to play from the beginning. Pause indicates playback has been paused and will resume from the last played moment. Play indicates playback will either start from the beginning or resume from the last played moment.

3.4 Save Functions

Different parts of the GUI can be saved using the Save menu (Fig. 1a). The entire GUI display can be saved as a JPEG or as a PDF. The user can choose the save location and name the file. Each individual graph can be saved as a PDF that can be infinitely zoomed to retain quality of data. The new distorted sound can be saved as a *.wav file to a user-defined location. Information about the distance from source and the ground type will be appended to the name of the sound file.

For ease of use, the user can also set the start path for various functions. SCAFFIP Data Files determines where SCAFFIP looks for the data files. Read Sound Files sets the start directory for loading sounds. Save Sound Files sets the start directory for saving sound files. Save Graphs sets the start directory for saving any JPEGs or PDFs of the GUI.

3.5 Undistort Function

Using the Function menu (Fig. 1a), the user can switch between distorting and undistorting a sound. Using the Distort functionality, the user uploads a sound file and then can hear the sound under certain conditions. With the Undistort functionality, the user uploads a sound file that was taken under nonideal conditions and can undo the atmospheric distortion to retrieve the undistorted sound file.

3.6 Distance Function

The distance function needed additional work, the program predicted results that were outside an acceptable range and returned values that were not a number. This function was therefore left out of the final product.

3.7 Testing the Program: Gathering Sound

Sound files were recorded at 5-m intervals up to 60 m away from the sound source. The atmospheric conditions at the time of collection are listed in Table 1 and the ground conditions are listed in Table 2. These parameters were inputs used by the SCAFFIP program to calculate the sound files.

Table 1 Atmospheric conditions

Variable	Value
Longitude	39
Latitude	77.2
Time	2100 hours
% Cloud cover	60%
Date	7/24/2014
Windspeed (mph)	0
Wind direction (knots)	0
Temperature (°C)	20
Humidity (%)	80%

Table 2 Ground conditions

Variable	Value
Source height (m)	0.5
Receiver height (m)	1
Ground type	Grass-covered pasture

The collected sound files were compared with the predictions by calculating the correlation coefficient between the Fourier transform of the collected and predicted signals. This would demonstrate how accurate the decibel attenuations of each frequency and distance that was measured.

The r values (Table 3) were relatively high for smaller distances and decreased as distances increased. At 10 m, the r value was 0.79 while at 60 m the r value was only 0.44. Additionally, because there were so many individual amplitude measurements made, the probability of these correlation values occurring by chance was practically zero. This shows that whatever caused the difference between the calculated and the actual distorted sound frequencies must have been a systematic difference and not due to random chance. However, to a human ear, the sound volumes are so low at distances greater than 20 m that the differences in frequency are not noticeable.

Table 3 Correlation between frequencies of predicted and actual distorted sounds

Distance (m)	r (Correlation coefficient)
5	0.787
10	0.7948
15	0.6925
20	0.7056
25	0.647
30	0.5787
35	0.5956
40	0.6064
45	0.4654
50	0.5058
55	0.4677
60	0.4393

The results of the attempt to calculate distance are listed in Table 4. The predictions appear extremely close for 5, 20, 30, and 40 m but are very far off for the other distances. For the distances whose calculated distance is listed as NaN (not enough numbers), the difference in frequencies were great enough that the projected distance was greater than 80 m and therefore outside of the matrix generated by the SCAFFIP. Because so many of the data could not generate numeric values, the function could not be completed as designed and was left out of this version of the program.

Table 4 Calculated distance compared with actual distance

Actual distance (m)	Calculated distance (m)
5	6.6436
10	48.6233
15	59.6017
20	22.9499
25	NaN
30	29.8868
35	NaN
40	45.3063
45	8.5006
50	NaN
55	NaN
60	NaN

4. Conclusion

The GUI was able to display a filter that could be used to allow the user to gain a better understanding of the atmospheric effects of sound propagation. The save functions also allow the user to compare individual data points later, as well as overall graphs, which makes the GUI useful for future research on sound distortion of specific files. The initial results reflect the expectation that sound will travel faster at night, indicating sound degradation will occur slower.

Several challenges remain in the simulation. Errors in the correlation values may be in the data collection or in the program. In the program, it is possible that the linear interpolation between points on the matrix was not an adequate approximation, which led to the error between the predicted and observed sounds. It is also possible that when data were collected, the effect of wind on the microphone caused static and other noises that introduced other signals to the original, meaning that it was not directly comparable to the original. As the distance increased, other noises would be more prominent in the recording and would lead to a decrease in correlation between the collected and predicted frequencies, which reflects the decreasing trend demonstrated in Table 3. The distance function could not accurately match a sound profile to its distance. This is a related problem to the lack of correlation between the predicted and collected data. It was difficult for the program to identify which distance profile best fit the input sound because there was not a very high similarity between the predicted and the actual signals. These issues can be addressed as work continues on the simulation.

5. References

1. Attenborough K. Sound propagation in the atmosphere. In: Rossing TD, editor. Springer handbook of acoustics. New York (NY): Springer; 2007. Chapter 4. p. 113–143.

2. Lamancusa JS. Outdoor sound propagation. State College (PA): Pennsylvania State University. ME 458 Engineering Noise Control; Fall 2000 [accessed 2015 Dec 18]. p. 10.1–10.19. http://www.me.psu.edu/lamancusa/me458/10_osp.pdf.

3. Noble JM. User manual for the Microsoft Window edition of the scanning fast-field program (WSCAFFIP) version 3.0. Adelphi (MD): Army Research Laboratory (US); 2003 Jan. Report No. ARL-TR-2696.

INTENTIONALLY LEFT BLANK.

Appendix. Distort Function Code, Listed by Functions in Alphabetical Order

This appendix appears in its original form, without editorial change.

```
function newSound=audioDistort(imptSound, fs, freq,
dB, axes1, axes2, axes3, humid, purpose,dBaxes)

[~,col]=size(imptSound);

[dftaudio, freqVec]= dftFreq(imptSound,
fs); %do the fft transformation
newdB=dBVec(freqVec,dB,freq,humid, purpose); %get the
decibel
attenaution matrix
%plot(dBaxes,freqVec,newdB); %plot the filter on the
axes passed in

dftNewLeft=getDistortFFT(newdB,dftaudio,1);
%distort the left side
newSoundLeft=(real(ifft(dftNewLeft))); %get the
signal for the left

if(col==2)
    dftNewRight=getDistortFFT(newdB,dftaudio,2);
%distort the right side
    newSoundRight=(real(ifft(dftNewRight))); %get
the signal for the right
    newSound=[newSoundLeft newSoundRight];
    %concatenate the two sides
together
else
    newSound=newSoundLeft;
end

graphAudioInfo(newSound,fs,axes1,axes2,axes3);
--
```

```
function dBVec=dBVec(freqVec,dB,freq,humid,purpose)
%freqVec is the vector with frequencies to match the
sound file
%dB is the read in dB attenuations
%freq is the read in corresponding frequencies
%humid is the humidity of the air
%purpose is either -1 or 1 and indictes whether the
sound is to be
%distorted or undistorted

%plot attempt using
interp1 function
if(purpose==-1)
```

```
    dBVec=purpose*interp1(freq,dB,freqVec,'linear',0.1
    );

elseif(purpose==1)

    %calculate
    slope for each
    bin
    tempdB=dB(2:end
    ); %offset dB
    deltadB=tempdB-dB(1:end-1); %calculate
difference in dB betweem bins
    tempFreq=freq(2:end); %offset freq
    deltaFreq=tempFreq-freq(1:end-1); %calculate
difference in freq between bins
    slopeMat=deltadB./deltaFreq;

    %calculate the
    number of bins
    binNum=length(f
    req)-1;

    %initialize all the expanded matrices for
    frequency, dB and slope expFreq=[];
    expdB=[];
    expSlope=[];
    %expand dB and freq to match the
    size of freqVec for k=1:binNum
    %run once for each bin
        binLength=getFreqIndex(freq(k+1),freqVec)-
getFreqIndex(freq(k),freqVec); %calculate the
length of the current bin
        expFreq=[expFreq,
repmat(freq(k),binLength,1)']; %expand
frequency to the length of the bin
        expdB=[expdB, repmat(dB(k),binLength,1)'];
        %expand dB to the
length of the bin
        expSlope=[expSlope, repmat(slopeMat(k),
        binLength, 1)'];
%expand slope to the
    length of the bin
    end

    %set the variables to their
    expanded selves dB=expdB;
    freq=expFreq;
    slopeMat=expSlope;
```

```
    %cut freqVec to appropriate
    length so it matches
    truncFreqVec=freqVec(1:length(dB
    ));

    %calculate dB for each
    frequency value
    dBVec=(truncFreqVec-
    freq).*slopeMat+dB;
    %}

    %extrapolate dB decrease so it covers the whole

audio file

freq=[1000,1600,2000,2500,3150,4000,5000,6300,8000,10

000,12500,16000,2
0000,25000,31500,40000,50000,63000,80000,100000]';
    shumidity=[10,20,30,40,50,60,70,80,90];
    %attenuations for various frequencies and
    humidities dbAtten=-1*[array of values];
    wantedHumid=str2double(humid); %get the
humidity column for the passed on humidity
    diffLength=length(freqVec)-length(dBVec);
%length that new freq matrix needs to be
    wantedFreq=linspace(truncFreqVec(end),freqVec(end)
    ,diffLength);
%generate the freq matrix for the higher dB
attenuations

extrapAtt=interp2(humidity,freq,dbAtten,wantedHumid,wa
ntedFreq,'linear
')';%interpolate to get the actual dB values and
invert so it matches dimensions
    dBVec=[dBVec extrapAtt]*purpose;
    disp('I am doing this based on humidity');
end
--



%does a fast fourier transform on a given audio
and returns the fft and the
%vector of correlating frequency values in Hertz
function [dftaudio, freq]= dftFreq(audio, fs)
```

```
%calculate basic spacing of the audio
file deltaT=1/fs; %defines the time
in between each sample
N=length(audio); %number of smaples
t=(0:1:N-1)*deltaT; %vector enumerating
each time index periodT=N*deltaT; %time
it takes to complete a period
deltaF=1/(periodT); %spacing on the
frequency scale
freq=(0:1:N-1)*deltaF; %vector enumerating each
frequency index

%do the
Fourier
transform
dftaudio=(f
ft(audio));
--
%given dB attenuation and the fft of an audio
clip, return the modified
%audio fft
--
```

findDistance

```
%read in SCAFFIP files
proptabPath=uigetdir('C:\Users\raglin_guest
2\Documents\Data Collection\proptab
files\Grass Jul 24','Select SCAFFP Data
Files');%get the folder containing the
SCAFFIP proptab files
[dB, freq, range]=readProptabData(proptabPath); %get
back the matrix

%Import the clean sound
[FileName,PathName]=uigetfile({'*.wav';'*.au'},'S
elect Clean Sound
File','C:\Users\raglin_guest2\Documents\MATLAB\Te
st Sounds\Voice Record Pro\');
pathName=strcat(Path
Name,FileName);
[cleanSound,fs]=wavr
ead(pathName);

for k=5:5:60
%Import the disorted sound
```

```
%[FileName,PathName]=uigetfile({'*.wav';'*.au'},'Selec
t Distorted
Sound
File','C:\Users\raglin_guest2\Documents\MATLAB\Test
Sounds\Voice
Record Pro\');
PathName='C:\Users\raglin_guest2\Documents\MATLAB\Test
Sounds\Voice
Record Pro\';
FileName=strcat(num2str(k),'m.wav');
pathName=strcat(PathName,FileName);
[distortSound,~]=wavread(pathName);


%find dB attenuation between the distorted and
clean files cleanDistance=.5; %the distance at
which the clean measurement was taken
[useAtten,dBAtten,dftFreq]=getdBAtten(cleanSound,disto
rtSound, fs);
%obtain the attenuations for corresponding frequencies

attens=interp1(freq,dB',400); %find the dB
attenuations for a frequency of 400
%attens=median(dB');
distance=interp1(attens',range,useAtten); %get the
distance for the found attenuation

disp([num2str(k),': ',num2str(distance),'m away.']);
%display results end

--
```

function newAmp=getDistortFFT(dBMat, fftAudio, channel)

```
dBMat=dBMat'; %reshape the dBMatrix to the

right dimensions

fftAudio=(fftAudio(:,channel)); %take one

side of the audio
phaseComp=angle(fftAudio); %get the phase of the
audio (so that it can later be recombined with the
magnitude

%convert dB attenuation into amplitude change
```

```
%use dB=20*log(value/ref) and solve for value
%use fftAudio for ref and dBMat for dB
fftAudio=(fftAudio(1:length(dBMat)));
phaseComp=phaseComp(1:length(dBMat));
fftAudio=abs(fftAudio);
newMag=dBMat/20;
newMag=10.^newMag;
newMag=newMag.*fftAudio;
%newAmp=newMag.*cos(phaseComp)+1i*sin(phaseComp);%com
bine new magitude and phase to get complex numbers
newAmp=newMag.* exp(1i*phaseComp);
--
```

```
function index=getFreqIndex(getFreq,freqVec)
deltaF=freqVec(2)-freqVec(1);
index=round(getFreq/deltaF);
--
```

```
%graphs freqency vs magntiude and puts it into given
plots

function graphAudioInfo(audio,fs, axes1, axes2, axes3)

%clip=audio(1:length(audio)); %select part
of the audio file clip=audio;
%clip=audio(1:length(audio))/10; %decrease volume by a
factor of 10

%calculate basic spacing of the audio
file deltaT=1/fs; %defines the time
in between each sample
N=length(clip); %number of smaples
t=(0:1:N-1)*deltaT; %vector enumerating
each time index periodT=N*deltaT; %time
it takes to complete a period
deltaF=1/(periodT); %spacing on the
frequency scale
freq=(0:1:N-1)*deltaF; %vector enumerating each
frequency index

%do the Fourier transform
dftclip=fft(clip);
dftclip=dftclip/length(dftclip);
```

```
%plot the raw sound as
time vs amplitude
plot(axes1,t,real(clip));
title('Time vs Amplitude'); xlabel('Time (sec)');
ylabel('Amplitude');

%plot a linear freq vs magnitude graph for
part of the data dispLength=10000; %length
for freq vs magnitude linear graph
freqDisp=freq(1:dispLength); %shorten
frequency scale
messDisp=dftclip(1:dispLength); %shorten
data plot(axes2,freqDisp,abs(messDisp));
%plot freq vs magnitude
%plot(freqDisp,abs(real(messDisp)));   %plot
freq  vs  magnitude  title('Linear  Freq  vs.
Magnitude');xlabel('Frequency        (Hz)');
ylabel('Amplitude');

%plot a semilog x graph of freq vs
magnitude for all data if 20000/deltaF<=N
    freqEnd=20000/deltaF; %index on the frequency
vector for 20000 Hz else
    freqEnd=N; %go to the highest frequency
end
freqStart=20/deltaF; %index on the frequency
vector for 20 Hz
graphFreq=freq(freqStart:freqEnd);
graphdft=abs(dftclip(1:freqEnd-
freqStart+1));
%graphdft=abs(real(dftclip(1:freqEnd-freqStart+1)));
semilogx(axes3,graphFreq,graphdft);
title('Semilogx Freq vs. Magnitude');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

end
--
```

function [dB,freq]=importFilter(dBaxes,dist,folderPath,purpose)

```
dist=str2double(dist);
[dB, freq]=readProptabData(dist,folderPath); %read in
the data from
SCAFFIP output
```

```
%plot the decibel attenuation
plot(dBaxes,freq,dB*purpose);
title('dB Attenuation');xlabel('Frequency
(Hz)');ylabel('dB');
--
```

```
function [t, freqAxis, ampAxis]= plotAmpFreq(handles,
audio, fs)
%calculate basic spacing of the audio file
deltaT=1/fs; %defines the time in between each
sample N=length(audio); %number of smaples
t=(0:1:N-1)*deltaT; %vector enumerating
each time index periodT=N*deltaT; %time
it takes to complete a period
deltaF=1/(periodT); %spacing on the
frequency scale
freq=(0:1:N-1)*deltaF; %vector enumerating each
frequency index

%do the Fourier transform
dftaudio=fft(audio);

%plot a semilog x graph of freq vs magnitude for all
data


if 20000/deltaF<=N
    freqEnd=20000/deltaF; %index on the frequency
vector for 20000 Hz else
    freqEnd=N; %go to the highest frequency
end
freqStart=20/deltaF; %index on the frequency vector
for 20 Hz


freqAxis=freq(freqStart:freqEnd);
ampAxis=abs(dftaudio(1:freqEnd-freqStart+1));

end

--
```

```
function [dB, freq]=readProptabData(dist,folderPath)
%filePath=uigetdir;
```

```
%[fileName,filePath,index]=uigetfile

fileSpecsName='proptab.pth';
fileDataName='proptab.ptd';
%% import spec document for SCAFFIP output
fileSpecsPath=strcat(folderPath,fileSpecsName);
%specify path to the spec file
specID=fopen(fileSpecsPath); %open the file
formatSpec='%s'; %read as a string
fileSpecs=textscan(specID,formatSpec,'delimiter','
\n'); %read specs line by line into a cell of
strings
fclose(specID);%close the file

row=str2num(fileSpecs{1}{12})+1; %get the number of
rows in the data file
column=str2num(fileSpecs{1}{14})+1; %get number of
columns in the data
file
%% import data document into dataMatrix
fileDataPath=strcat(folderPath,fileDataName);
%specify path to the data file
dataID=fopen(fileDataPath); %open the
file formatData='%f'; %read as floats
fileData=textscan(dataID,formatData); %read data
number by number into a cell containing a one
dimensional float array
fclose(dataID); %close file

dataMatrix=fileData{1}; %get element out of
cell; it's a one dimensional float array
dataMatrix=reshape(dataMatrix,column,row)'; %reshape
matrix into 2
dimensional matrix of correct dimensions
%%
stepRange=dataMatrix(3,1)-dataMatrix(2,1);
%dist=50;
getRow=round(dist/stepRange+1);
%get desired part of data Matrix; first and last rows
%dBMatrix=dataMatrix(1:row-1:row,:);
freq=dataMatrix(1,:);
%dB=dataMatrix(row,:);
dB=dataMatrix(getRow,:);
dB(1)=0;
--
```

Function writeDistortedSound(sound,fs,fileName,dist,ground)

```
ground(ground==' ') = '_'; fileName=fileName(1:end-4);
newName=strcat(fileName,'_',dist,'m_',ground,'.wav'); wavwrite(sound,fs,newName);
msgbox(strcat('Successfully saved to: ',newName));
```

Bibliography

Alberts WCK II, Coleman MA, Noble JM. Characterization of acoustic ground impedance at Blossom Point Research Facility. Adelphi (MD): Army Research Laboratory (US); 2010 Sep. Report No. ARL-TR-5352.

MathWorks. Documentation: Signal Processing Toolbox Functions. . Natick (MA): The MathWorks, Inc. [accessed 2014 Nov 12]. http://www.mathworks.com/index.html?s_tid=gn_logo.

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
3 (PDF)	DIRECTOR USARL RDRL CII B A RAGLIN RDRL CIE S J NOBLE S WAGNER

INTENTIONALLY LEFT BLANK.